Using Timer1 Functionality with Pulse-per-second signal (PPS) May 26 23

Jun ha Kim

Advisor: Professor Raul Armendariz

Pulse-per-second signal (PPS) explained

- Pulse-per-second signal (PPS): “Accurate” electronic signal that repeats once per second. Derived from the Adafruit GPS signal mounted on the breadboard
- PPS signal itself lasts 50-100ms and pulses high (3.3V)
- Signal is typically used for time synchronization purposes; documentation states that PPS pulses emitted have an accuracy of 100-300 nanoseconds
- Note: given that we use Serial Monitor to capture data the underlying time is derived from the system i.e. Windows

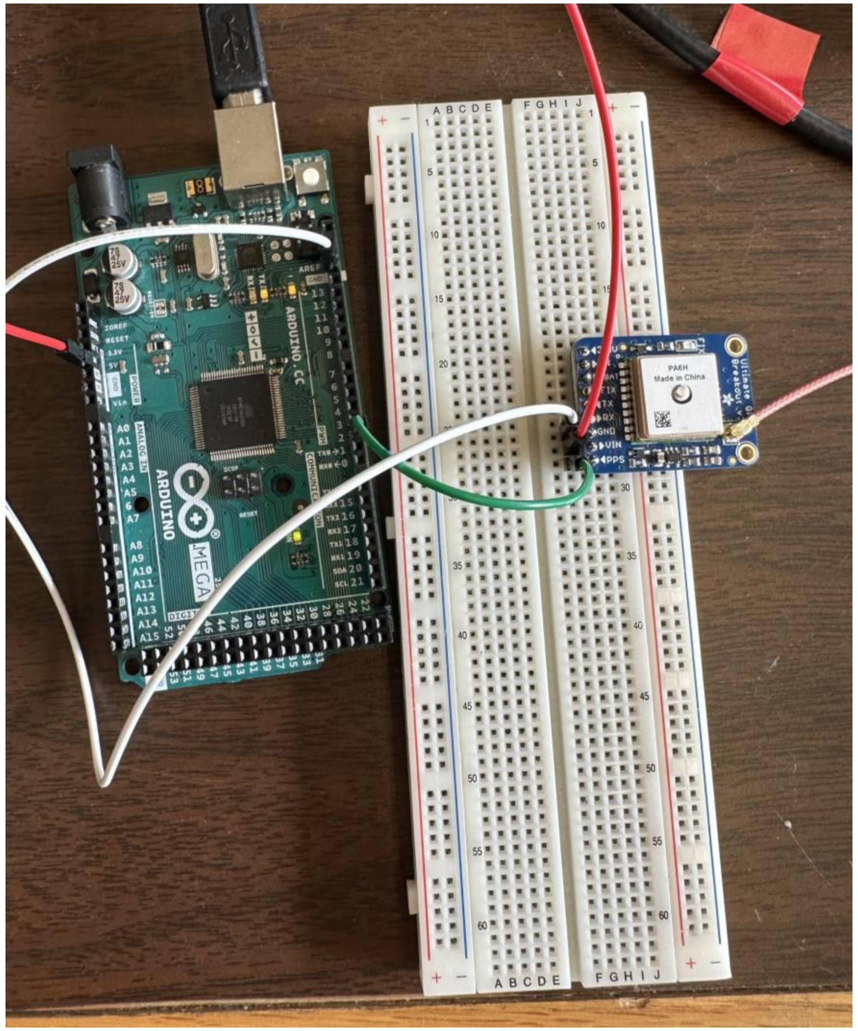
PPS Program Objectives

- Every time the PPS pulse rises, the PPS() function is called. Timer1 begins incrementing to 65,535. Each time the Timer1 overflows, the “overflows variable” increments. This latter piece of data is printed
- This program will print the number of Timer1 overflow occurrences between two PPS signals (prints every second)
- It will also print the number of clock cycles that have occurred in the span of 1 second. This is derived by multiplying the Timer1 overflow occurrences by 2^{16}
- Overflow occurrences variable is reset to 0 after 1 second; each line denotes overflow occurrences between two distinct PPS signals

PPS Calculation Results

- // No actual results collected
- Given that a PPS signal has a frequency of 1 second

Arduino Setup



- Items required:
  - Arduino Mega 2560
  - Adafruit GPS sensor mounted on breadboard
- Wire together:
  - Arduino 5V to GPS VIN
  - Arduino GND to GPS GND
  - Arduino PPS to PWM2

Pin Number Names, Global Variables

```
#define pPS 2 // PPS signal pin
unsigned long overflows = 0; // Note: if the program does not work, try
setting this to an integer that is not 0 e.g. 5
unsigned long overflowCount = 0;

ISR(TIMER1_OVF_vect) // Interrupt Service Routine
{
  overflows++; // Increases the "overflows" variable by 1
}
```

Setup() Function

```
void setup(){
   Serial.begin(115200);
   delay(1000);
   pinMode(pPS, INPUT);
   TCCR1A = 0; // Sets entire TCCR1A--Timer1 Control Register A--to 0
   TCCR1B = 0; // Timer 1 Control Register B set to 0 (The physical address of timer1)
   TCCR1C = 0; // Timer 1 Control Register C set to 0
   TCNT1 = 0; // Initialize timer/counter 1's value to 0
   TIMSK1 = _BV(TOIE1); // Timer/Counter1's interrupt mask register; TOIE1 is the timer/Counter1
overflow interrupt enable
   TCCR1B = 1; // Timer 1 Control Register B set to 1
   attachInterrupt(digitalPinToInterrupt(pPS), PPS, RISING);
}
```

PPS() Function

```
void PPS() {
   unsigned int temp = TCNT1; // Only positive integers are required
   Serial.print("Overflow Count: ");
   Serial.println(overflows);
   Serial.print("Overflows x 2^16: ");
   Serial.println(overflows<<16); // i.e. overflows * 2^16 or number of clock
cycles between two PPSs
   Serial.print("TCNT1: ");
   Serial.println(temp);
   overflows = 0; // Resets the overflows and overflows<<16 variable to 0
}
```

Loop() Function

```
void loop(){ // The contents of this function are not strictly necessary
  while(digitalRead(pPS) == HIGH){
  }
  while(digitalRead(pPS) == LOW){
   // Printing zeroes continuously serves no purpose
  }
}
```

- Data is printed to the Arduino Serial Monitor